A GA-Based Scheduling Method for FlexRay Systems

Shan Ding, Naohiko Murakami, Hiroyuki Tomiyama and Hiroaki Takada
Graduate School of Information Science,
Nagoya University
Furo-cho, Chikusa-ku, Nagoya 464-8603, Japan
{ding, murakami, tomiyama, hiro}@ertl.jp

ABSTRACT

An advanced communication system, the FlexRay system, has been developed for future automotive applications. It consists of time-triggered clusters, such as drive-by-wire in cars, in order to meet different requirements and constraints between various sensors, processors, and actuators. In this paper, an approach to static scheduling for FlexRay systems is proposed. Our experimental results show that the proposed scheduling method significantly reduces up to 36.3% of the network traffic compared with a past approach.

Categories and Subject Descriptors

D.4.7 [**Organization and Design**]: Real-time systems and embedded systems; D.4.1 [**Process Management**]: Scheduling; F.2.2 [**Nonnumerical Algorithms and Problems**]: Sequencing and scheduling

General Terms

Algorithms, Theory

Keywords

distributed embedded systems, FlexRay, genetic algorithm

1. INTRODUCTION

The FlexRay system[1] is a communication system developed for the next generations of automobiles by a consortium founded in 2000 by BMW, DaimlerChrysler, Motorola, and Philips Semiconductors. The core of the FlexRay system is the FlexRay communication protocol. It has been designed for high data transmission rates required by advanced automotive control systems.

The FlexRay system is a time-triggered architecture providing a computing infrastructure for the design and implementation of dependable distributed embedded systems. The communication in this architecture based on a fault-tolerant time-triggered protocol.

In the FlexRay protocol, media access control is based on a recurring communication cycle. Within one communication cycle FlexRay offers the choice of two media access schemes. One is

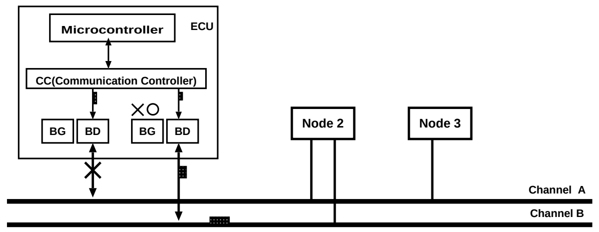

Figure 1: The node architecture and dual bus topology configuration

a static time division multiple access(TDMA) scheme, and the other is a dynamic mini-slotting based scheme. The portion of the communication cycle where the media access is controlled via a TDMA scheme called static segment.

In FlexRay systems, a structure used by the communication system to exchange information within the system is called a frame. Frame is sent when the slot number is corresponding to frame ID. In the static segment, a periodic transmission of the fixed length data is guaranteed. Static slot messages can be protected by Bus Guardian (BG). From above characteristics, an application messages had better be allocated to static segments in order to ensure that it can meet its time constraints.

For various sensors, processors, and actuators with different execution period, it is necessary to develop an efficient scheduling method for static segment of communication cycle in the FlexRay system. An optimization procedure based on a Simulated Annealing (SA) has been proposed by Murakami et al [5]. In this paper, an approach to static scheduling for FlexRay systems is proposed. The advantages of GA-based approach depends heavily on how well the various components of GA incorporate the salient features of the problem under consideration. To evaluate the effectiveness of this approach, we have chosen a representative safety critical application to simulate as case study. Our experiments show the GA can be applied to such kinds of application to find a schedule better than SA approach.

2. SYSTEM ARCHITECTURE

The FlexRay system architectures consisting of nodes are connected by broadcast communication channels. The FlexRay protocol is a dual channel protocol. FlexRay systems can be configured as single-channel or dual channel bus network, a single-channel or dual-channel star network, or in various hybrid combinations of bus and star topologies. Figure 1 shows one kind of topology configuration of the communication network as a dual bus.

In the FlexRay protocol, media access control is based on a recurring communication cycle. Figure 2 shows one complete instance of the communication structure of a communication cycle. Cycle counter, the number of the current communication cycle, ranges from zero to 63.

Because the communication channel is a broadcast channel, a message sent by a node is received by all the other nodes.

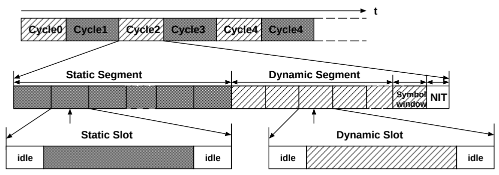

Figure 2: The communication cycle

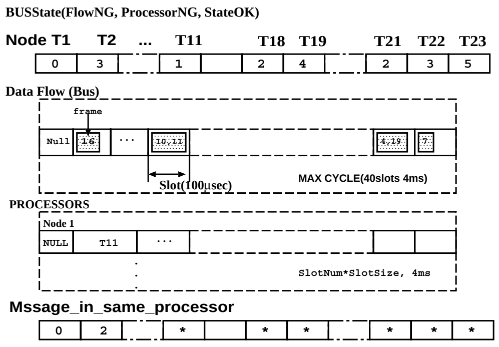

Figure 3: An individual

Node can transmit only during a predetermined time interval, so-called TDMA. In a slot, a node can send several messages packaged in a frame. The sequence and length of the slots are the same for all TDMA rounds. However, the length and contents of the frames may change. Each node consists of a microcontroller(host), a communication controller(CC) a bus driver(BD), and bus guardian(BG).

The periods of tasks and messages are assumed to be the value 2^n(n = 0,1,2,...) times at FlexRay communication cycle. The main reason for this assumption is to avoid the time violation between messages. For example, if m_1 and m_2 have periods of 2, 5 respectively, it can not avoid a time violation at communication cycle. In an actual design, it is necessary to adjust the communication cycle and the period of the application in FlexRay systems.

3. PROBLEM DESCRIPTION

This paper addresses the same problem as proposed in [5]. In order to indicate our algorithm, we give a brief description of the problem as follows. An application consists of a set of task graphs $\{G_i = (T_i, E_i)\}$, where T_i is a finite set of task vertices(i.e., nodes) , E_i is a finite set of message edges representing connetions between these nodes. Each node T_i is allocated to a uncertain processor, and has a known worst case execution time C_i, a period T_i, a deadline D_i. Output is a schedule comprising identical bus.

As the primary objective, we construct a schedule meeting all deadlines of task groups and performance goals of the embedded application. The secondary objective is that the schedule can optimize communication buses to minmize hardware cost. Since bin-packing problem is known as $NP-hard$ [3], in this paper, we propose a GA-based scheduling method.

Constraints

Response Time (RT): To the output task processes, messages are sent by the input task processes through the communication routes. RT is the time from beginning of all input task processes that inflected by the output task processes, until end of the output task processes.

Freshness Time (FT): Messages are sent by the input task processes through the communication routes. FT is the time from beginning of input task processes until the end of all output task proceesses that be inflected by these messages.

Maximum Response Time: The maximum response time of all input tasks processes.

Maximum Freshness Time: The maximum Freshness time of all output tasks processes.

Based on above preliminaries, four time constraints are defined as follows;

Response Constraint: For certain route, the maximum response time must be less than a given time.

```
Generate Individual()
{
    for each edge e_i in edge set E_i;
    do {
        if (two vertexes of e_i in difference processors){
            Select a slot in dataflow randomly;
            while (unsatisfy contraints condition){
                Slip one slot;
                if (dataflow is enough){
                    BUSState = FlowNG;
                    break; }
            }
            if (satisfy contraints condition){
                Insert the edge into dataflow;
            }
        }else{
            write message_in_same_processor;
        }
    }while(Ei != ϕ)
    scheduling for each processor;
}
```

Figure 4: Algorithm for generating an individual

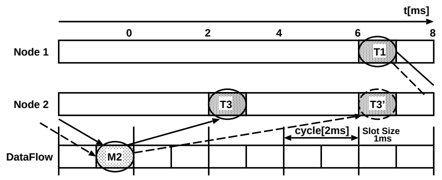

Figure 5: The example for calculating freshness constraint

Freshness Constraint: For certain route, the maximum freshness time must be less than a given time.

Synchronous Input Constraint: To these routes which have the same output task, the maximum difference of feshness time of these routes must be smaller than a given time.

Synchronous Output Constraint: To these routes which have the same input task, the maximum difference of response time of these routes must be smaller than a given time.

In this paper, we assume that all the constraints are equal to deadline of task graph. Besides above time constraints, we define another constraint by considering character of static segment.

Slot Redundancy: The number of the slot not used at the end of the communication cycle continuously expresses the degree of empty slots in the schedule. The larger number unused slots is, the higher the slot redundancy is.

4. GENETIC ALGORITHM FOR SCHEDULING PROBLEM

Our GA requires the definition of a set of genetic operations and an evaluation function as follows:

4.1 Coding Scheduling Individual

Figure 3 shows the structure of an individual. A string is used for showing which processor the node belongs to. The length of this string is the number of nodes in the task graph. The ith number in the string expresses that the ith node belongs to processor j. Another string is used for showing the messages (i.e., edges) in the communication cycle. We assume that the slot size is 32 bytes in this paper. If the communication cycle time is Δ_{cycle} and each transmission slot time is Δ_{slot}, the number of slots can be calculated as $\Delta_{cycle}/\Delta_{slot}$. As mentioned above, a structure called frame can load messages in the dataflow (bus). Several messages can be assembled into one frame when they are sent by the same node and sum of their sizes is not larger than the slot size. Several strings are prepared for scheduling of processor to record when and how long the nodes execute. The individual states include three states, processorNG, busNG, and stateOK.

The process of generating individual is shown in Figure 4. The process of message transmission can be considered two portions, i.e., send and receive. As a synchronous algorithm, the sending node is executed before dataflow (bus) slot number(i.e., time) in the processor which it belong to, and receiving node is executed after dataflow (bus) slot number (i.e., time) in the processor which it belong to. When there is no time to execute in the processor, the individual state will become processorNG state. When the message is transmitted in the same processor, it will not display in the dataflow. Finally, based on the node order appeared in the routes, we check the lost nodes in the processor strings. If there

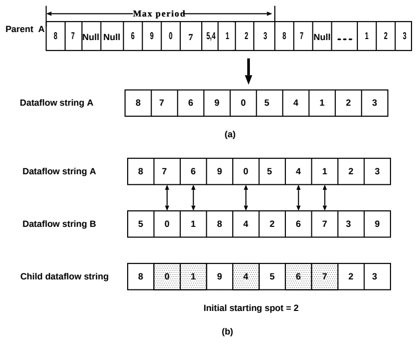

Figure 6: The crossover operation

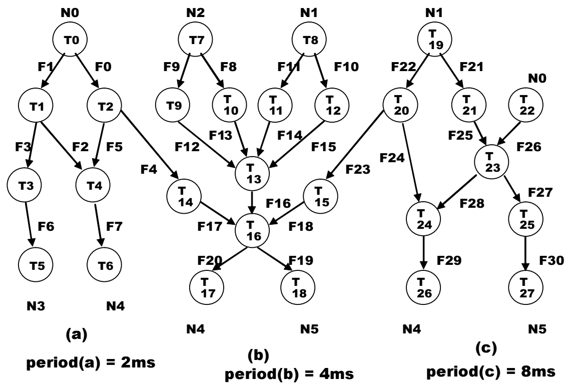

Figure 7: A safety critical application model

task	$e(\mu\text{sec})$	task	$e(\mu\text{sec})$	Edge	Size	Edge	Size
T0	150	T14	300	E0	12	E15	10
T1	350	T15	200	E1	12	E16	12
T2	200	T16	400	E2	20	E17	12
T3	250	T17	350	E3	12	E18	12
T4	300	T18	400	E4	20	E19	10
T5	200	T19	400	E5	12	E20	12
T6	400	T20	300	E6	12	E21	20
T7	300	T21	600	E7	10	E22	20
T8	350	T22	350	E8	12	E23	12
T9	400	T23	800	E9	10	E24	20
T10	250	T24	300	E10	10	E25	12
T11	400	T25	400	E11	12	E26	20
T12	300	T26	300	E12	20	E27	10
T13	400	T27	400	E13	12	E28	22
				E14	12	E29	20
						E30	12

Table 1: The execution time of nodes and the message sizes

Slot	Edge	Slot	Edge	Slot	Edge	Slot	Edge
0	4	20	20	40	4	60	20
1	9, 13	21	24	41	9, 13	61	null
2	3, 26	22	3	42	3	62	3
3	11, 15	23	28, 30	43	11, 15	63	null
4	17	24	null	44	17	64	null
5	7	25	7	45	7	65	7
6	23	26	null	46	null	66	null

Table 2: The network traffic of the optimum schedule

are lost nodes, it is necessary to schedule the lost nodes in their processors that they belong to. We insert lost node in the front of the node that appears to the individual by the same route.

4.2 Generate initial generation

Individuals are generated randomly as many as population size. The operation is performed as described elsewhere [2] in details.

4.3 Evaluation and Optimization Strategy

Our evaluation function captures the "degree of schedule" for a certain individual. When an individual is in stateOK state, we calculate the optimization cost value. According to definition of RT/FT, the routes which only include same period nodes or from longer period nodes to shorter period nodes must be calculated by response constraint for measuring the synchronous input/output constraints. However, when the routes is from shorter period nodes to longer period nodes, it is necessary to use the freshness constraint for measuring the synchronous input/output constraints.

If the route cost is larger than deadline, we give a penalty as the following formula.

$$P = \sum_{i=1}^{n} f_i = \begin{cases} \dfrac{R_i}{D_i}, \ if\ R_i \le D_i \\ \dfrac{(R_i - D_i)}{D_i} * \alpha + \beta, \ if\ R_i > D_i \end{cases}$$

f_i is the cost value of ith routes, R_i is the execution time of ith route which will be calculated by freshness time constraint or response time constraint. D_i is the deadline of the ith route. α and β are coefficients. We set α and β to 10 and 2, respectively.

In addition to the slot redundancy, the individual fitness value is calculated as follows,

$$Cost = P * w_1 + N_{slot} * w_2$$

N_{slot} is slot redundancy, w_1 and w_2 are coefficients. We set w_1 and w_2 to 1 and 0.5, respectively.

Figure 5 shows a freshness time constraint calculating case. The route can be expressed as T1 $\rightarrow$ m2 $\rightarrow$ T3. Task $T1$'s period and T3's period are 8ms and 4ms respectively. There are two routes from input task $T1$ to output task $T3$. They are expressed by solid lines and dotted lines. The cost values(i.e., time) of solid lines and dotted lines are 5 and 9 respectively. As a result, the maximum freshness time of this route is 9. The response time constraint can be calculated similarly, but depending on the execution timing of the input tasks instead.

4.4 Selection

As described above, all individuals in the population are sorted out according to their fitness, so the first individual is the best in this generation. The operation is performed as described elsewhere [2] in details.

4.5 Crossover

The crossover operation is held between two parents. If the two strings that show to which the node belongs are the same in two individual parents, we conduct crossover operation by randomly choose one of the three suboperations. One is merge the two messages in one slot in child individual. The second is inserting a message at the slot whose number that is equal to the average value of two parents' in child's string. The last one as shown in Figure 6 is to find a exchange sets in two parents and exchange genes in the child individual.

4.6 Mutation

The exchange mutations are defined as follows,

- Search for the slots unsatisfying constraints. Suppose that there are two messages unsatisfying constraints message$_a$ and message$_b$.
- If the period of message$_a$ is longer than that of message$_b$, then message$_b$ occupies the slot of message$_a$. Message$_a$ must find its new slot by scanning the static segment.

  If the period of message$_a$ is shorter than that of message$_b$, then message$_b$ must find its new position by scanning the static segment.
- If there is no slot unsatisfying constraints, randomly select two slots in static segment, exchange them in each cycle.

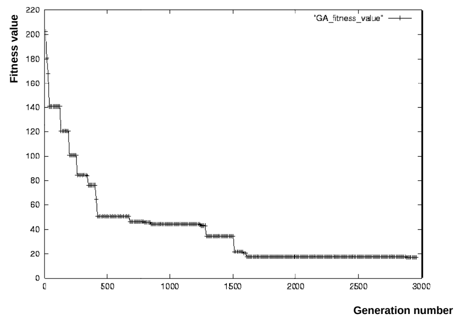

Figure 8: The relation between the best fittness value and the generation number

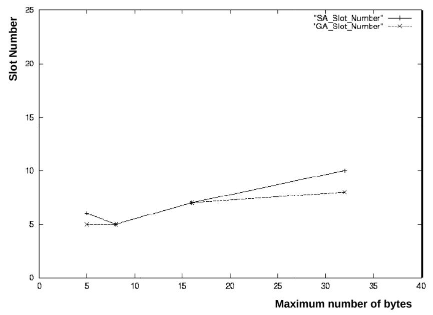

Figure 9: The relation between message size and number of slots

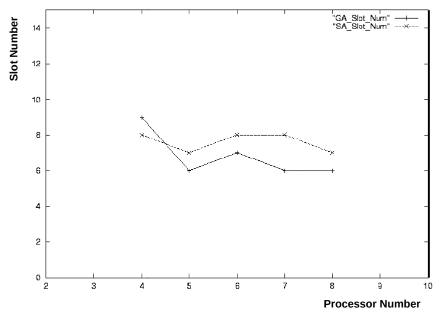

Figure 10: Maximum number of processors

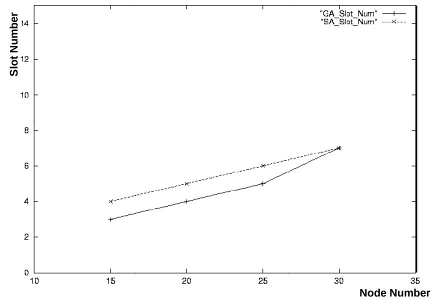

Figure 11: Maximum number of nodes

4.7 Differentiation of the same individual

After crossover or mutation, we sort the individuals. If the same individual is found, then we operate mutation to it in order to generate a new different individual.

5. EXPERIMENTS

To demonstrate the effectiveness of the GA-base scheduling algorithm, we conducted a set of experiments. We assume the FlexRay communication protocol having a bandwidth of 250kb/s and a bandwidth of 100μsec for the transmission slots. In this paper, we assume that the deadline of the node and edge is equal to the period of the node and edge. The developed GA is implemented using C language on the Linux.

A safety critical application with hard real-time constraints, to be implemented on a FlexRay based architecture, includes a vehicle adaptive cruise controller (ACC), electric power steering (EPS), and traction control (TC) as detailed in [4]. Considering the synchronous input/output constraints, we model the application as Figure 7. Task groups a, b and c have periods of 2ms, 4ms and 8ms, respectively. The Edge F4 and F23 have periods of 4ms, 8ms respectively. There are six processors available to allocate the nodes in this application. In Figure 7, the nodes labeled Ni, means that this node is allocated to ith processor. The other nodes can be allocated to any processor freely.

Table 1 summarizes the execution time of nodes and the various message attributes affecting network topology generation. The names of nodes and edges are in column "task" and column "Edge", respectively. The column "$e(\mu\text{sec})$" and column "Size" show the execution time (μsec) of nodes and message size (in byte) of edge, respectively.

Parameters for GA were set as follows. Population size was 28, generation number was 3000, optimum value was 17.09 (taking 1888 seconds of optimization time). Table 2 shows the network traffic of an optimum schedule. The Relationship between the best fitness value of generation and generation number is shown Figure 8.

In table 2, column "Slot" and column "Edge" show the slot number and edge name. Null means there is no edge in the slot. Table 2 shows the minimum number of slots can be used for schedule is 7. In [5], the minimum number of slot can be used for schedule was 11. This experimental result shows that the proposed scheduling method significantly reduces up to 36.3% of the network traffic compared with the SA approach. This schedule also can meet all deadlines of the task group a, b and c.

Next, we tested our algorithm with respect to the maximum message size allowed. For the results depicted in Figure 9 we have assumed the maximum message size as 5, 8, 16 and 32 bytes. Figure 9 shows that the number of slots used for schedule decreases with the decrease of the maximum number of bytes in a message.

We changed the graph for general experimental purpose. We considered four graph architectures consisting of 15, 20, 25, and 30 nodes. Execution time, periods and message size were assigned randomly within certain intervals. Figure 11 shows the number of slots used for schedule satisfying the constraints and deadlines. Good results were obtained by both algorithms. However, GA used a fewer number of slots than SA.

Moreover, we have assumed that 28 nodes were allocated to 4, 5, 6, 7 and 8 processors in a graph respectively. Figure 10 shows that GA is better than SA [5] at almost all kind of conditions. Experimental results show that our developed method is able to efficiently produce good quality results.

6. CONCLUDING REMARKS

In this paper, an approach to static scheduling for FlexRay systems is proposed. Some genetic operations are proposed in the GA. The optimization is performed on reducing the network traffic while meeting deadlines and satisfying the constraints which have been identified. A safety critical application that includes ACC, EPS and TC, has been studied in this paper. Our experimental results show that the proposed scheduling method significantly reduces up to 36.3% of the network traffic compared with a past approach. The effectiveness of the developed GA is confirmed by the experiments.

ACKNOWLEDGMENTS

This work is partially supported by JSPS postdoctoral fellowship for foreign researchers (No. P04284). The authors are also grateful to the industrial partners at Toyota Motor Corporation for their valuable feedback during this work.

7. REFERENCES

[1] J.Berwanger et al. FlexRay - the communication system for advanced automotive control systems. In *Proc. SAE World Congress*, Paper: 2001–01–0676, 2001.

[2] S. Ding and N. Ishii. Determining feature weights of pattern classification by using rough genetic algorithm with fuzzy similarity measure. *Journal of Japan Society for Fuzzy Theory and Systems*, 14(3):310-319 2002.

[3] D. S. Johnson. Fast algorithm for bin packing. *Journal Computer and System Sciences*, 3(2):272–314, 1974.

[4] N. Kandasamy, J. P. Hayes and B. T. Murray. Dependable communication synthesis for distributed embedded systems. In *Int'l Conf. on SAFECOMP*, LNCS 2788, pp.275–288. Sept. 2003.

[5] N. Murakami, S. Iiyama, H. Takada, M. Kido and I. Hosotani. A static scheduling method for distributed automotive control systems. *IPSJ SIG Technical Reports*, 3(27): 2005. (in Japanese).